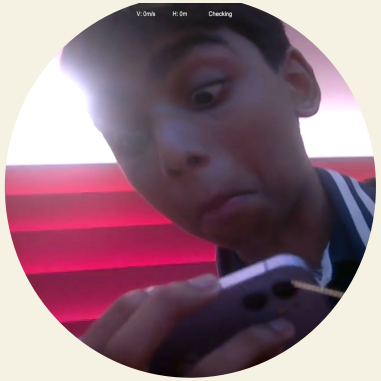
Lesson 6

1/2 of the way through the Swift course!

Your iOS teachers



Joshua Sheen
iOS teacher and CTO
Sec 2 Student
joshua_sheen@s2024.ssts.edu.sg

Contact:
DO NOT SPAM WE WILL COME AFTER YOU

Use it if you have questions. It may take a few working days (If we don't reply within 5 days then email us again) but we should be able to help.

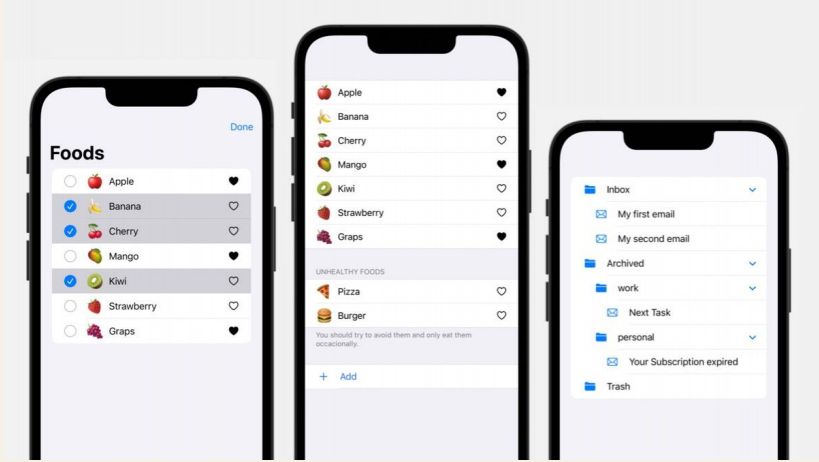
Kesler Ang
iOS teacher and ACTO
Sec 3 Student
kesler_ang_kang_zhi@s2023.ssts.edu.sg

Table of contents

01

List

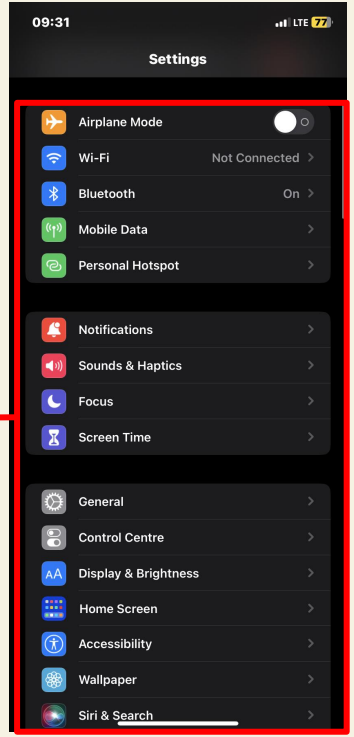
A List of things

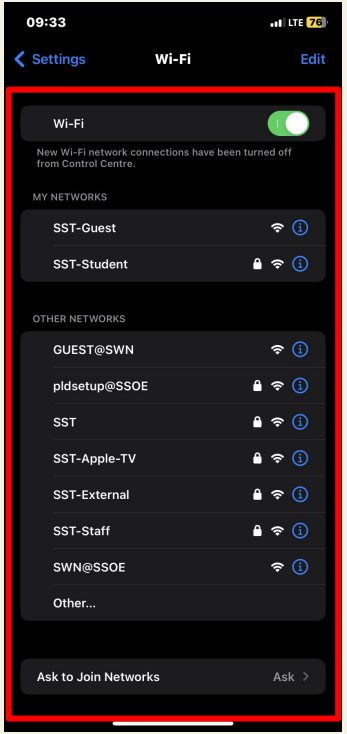


List

1. List is a “see” code
2. Commonly used to display data in a “table” form
3. It places text or variables into a list.

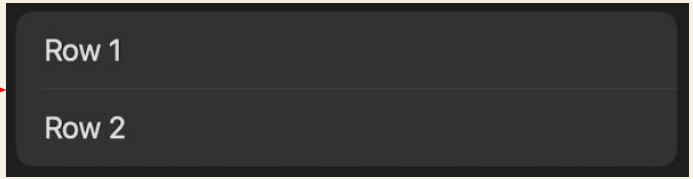
A list





List in SwiftUI

```
List {
    // each row content
    Text("Row 1")
    Text("Row 2")
}
```



List - Buttons

You can also put buttons in lists!

In your list brackets, state how you would usually create a button.

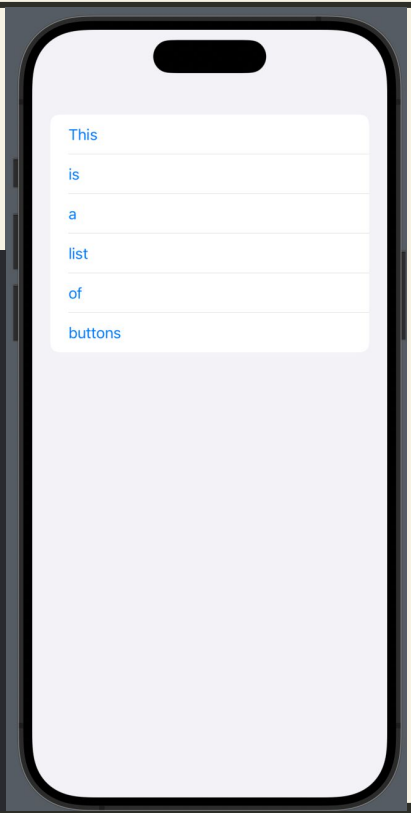
Declare your button under list

And, it is that simple!

```
struct ContentView: View {
    var body: some View {
        VStack {
            List{
                Button{

                } label: {
                    Text("A")
                }
                Button{

                } label: {
                    Text("List")
                }
            }
        }
    }
}
```



ForEach

1. ForEach is a “see” code.
2. Loops through an array and displays something (“see” code) for each element in the array.
3. Usually used with lists to make it easier to display information

For loops but it does “see” code on repeat and not “do” code on repeat.

We will use **forEach** to list down text in an array in the **list**

ForEach in SwiftUI

```
10 struct ContentView: View {
11     var shoppingCart = ["apples", "milk!!", "pear"]
12
13     var body: some View {
14         List {
15             ForEach(shoppingCart, id: \.self) { item in
16                 Text(item)
17             }
18         }
19     }
20 }
```

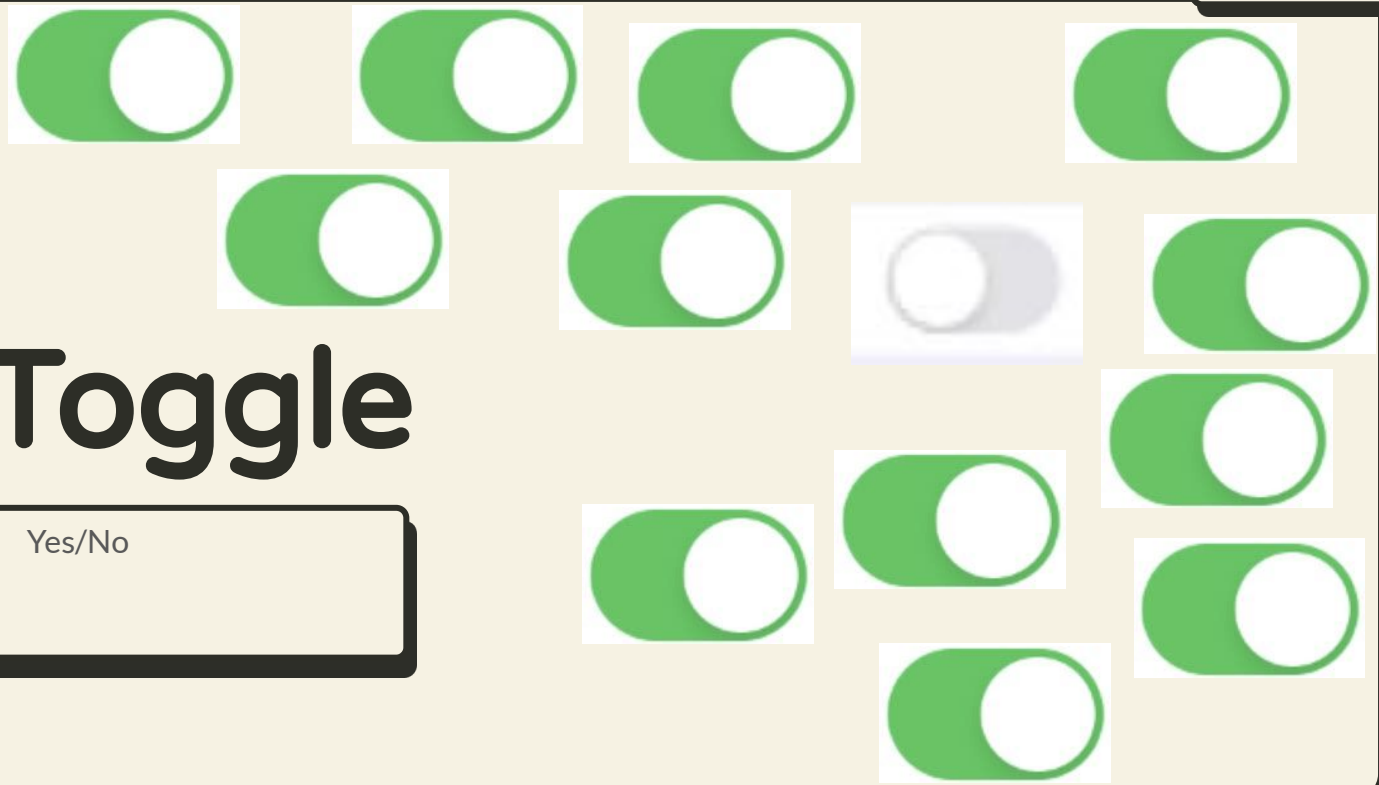
temporary variable to store the element that it is looping through

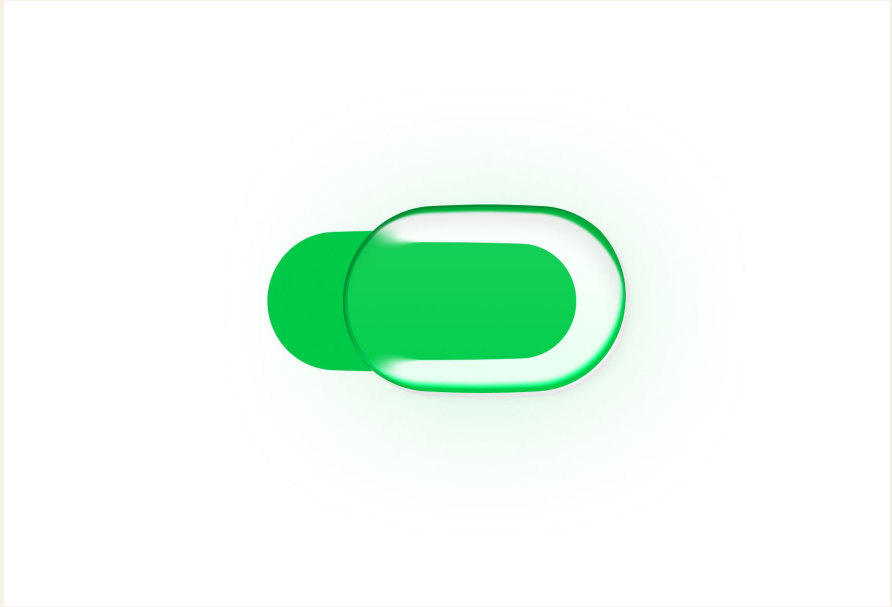
sets the id of each element to itself (so that swift knows that each element is unique)

02

Toggle

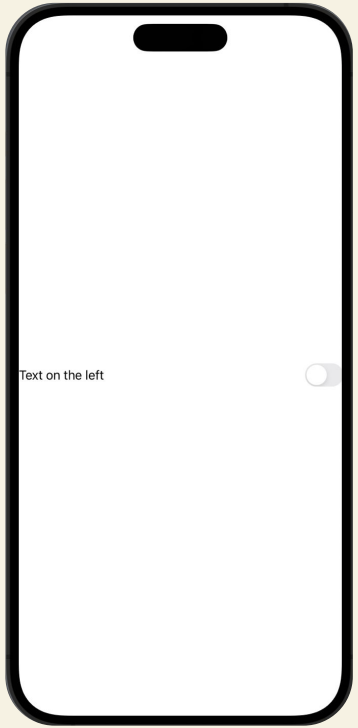
Yes/No





Toggle - the code

```
10
11 struct ContentView: View {
12
13     @State var isOn = false
14
15     var body: some View {
16         VStack {
17             Toggle("Text on the left", isOn: $isOn)
18         }
19     }
20 }
21
```



Toggle - how it works

1. What the Toggle( ) code does:
    a. Sets a boolean value from **true to false** or **false to true** when it is pressed
    b. Hence it needs a binding, so that Toggle is able to change the value of the variable

2. Note: there is .toggle() and Toggle(), they are DIFFERENT THINGS.

Mini Challenge 1

Make a list of text of a grocery list, and use toggle to switch on if the value of the items is true. Make each item a variable, e.g: var apple = true

var appleString = “Apple”

Integrate list with toggles, instead of buttons or text. ~25 mins

For those who wants a harder challenge (not compulsory),

You can instead use forEach to take variables from an array and use it

03

Stepper

Up or down?

Stepper - how it works

1. What the Stepper( ) code does:
    a. The first parameter is the text that shows up on the left
    b. Changes the value of $val by +1 or -1 when the buttons are pressed
        i. Hence, $val needs to be a binding so Stepper( ) can change the value

Stepper - change step?

1. **Step** is the value increase or decrease whenever the buttons are pressed
2. Default is 1

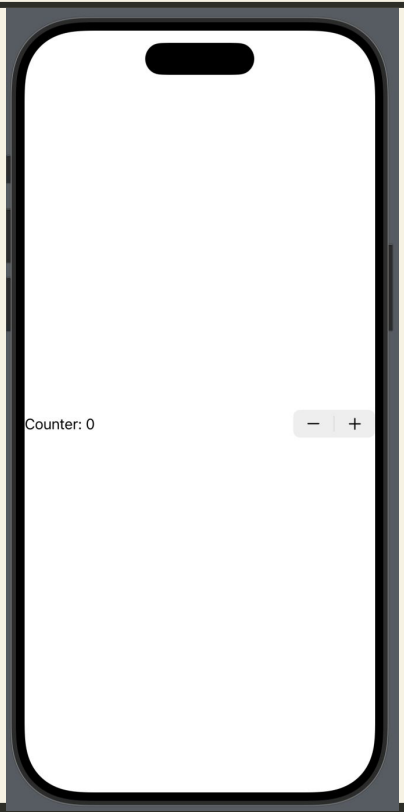
```
Stepper("Counter: ", value: $val, step: 3)
```

Stepper - the code

```swift
import SwiftUI

struct ContentView: View {
    @State var val = 0
    var body: some View {
        VStack{
            Stepper("Counter: \(val)", value: $val,
                step: 1)
        }
    }
}

#Preview {
    ContentView()
}
```



04

Spacer

Greedy View for the sake of styling

Spacer( )

1. Most SwiftUI components only take up as much space as they need
2. Spacers take up **as much space as they can**
3. Used for aligning a SwiftUI component
    a. E.g. to align something to the right, put it in a HStack with the Spacer( )
4. Not to be confused with .padding() (some might.)

Aligning things with Spacer()

```
10
11 struct ContentView: View {
12     var body: some View {
13         VStack {
14             Spacer() // Push to the bottom
15             HStack {
16                 Spacer() // Push to the right
17                 Text("Align to the right")
18             }
19             .background(Color.teal)
20             HStack {
21                 Text("Align to the left")
22                 Spacer() // Push to the left
23             }
24             .background(Color.red)
25         }
26         .padding()
27     }
28 }
29
```

1

2

3

05

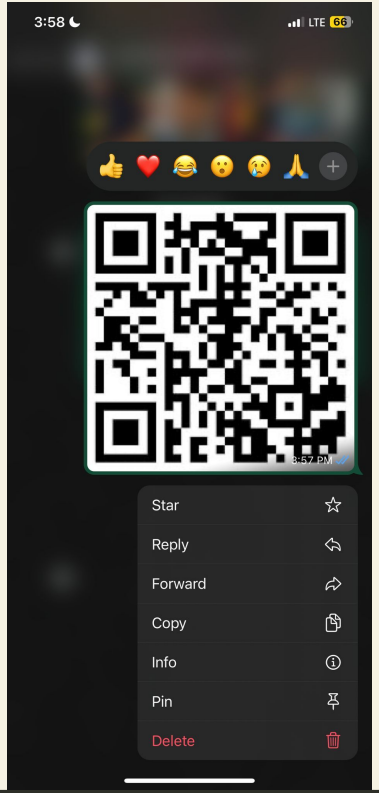
Context Menu

Hold and you get a list of buttons

contextMenu

contextMenu is popup buttons when you hold on a button, essentially a menu/list of buttons after holding another button or text.

E.g. This image—------------------------------>



Applications:

- Quick actions
- More functions in that button

contextMenu - quick action

You can use this code to do a quick action on a contextMenu button. You can do anything with quick actions. Let your imagination run wild!

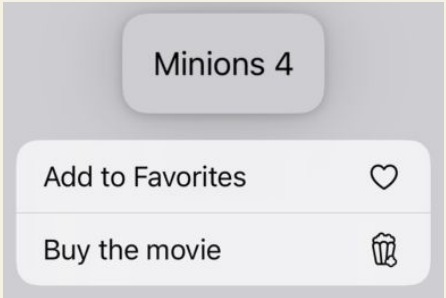
```
.contextMenu {
    Button("Cut", action: cut)
    Button("Copy", action: copy)
    Button("Paste", action: paste)
}
```

contextMenu - function

You can use this code to assign a function to a contextMenu button.

```
Text("Minions 4")
    .padding()
    .contextMenu {
        Button {
            // Add this fantastic movie to TedGPT's list
                of favorites.
        } label: {
            Label("Add to Favorites", systemImage:
                "heart")
        }
        Button {
            // Make user buy the movie
        } label: {
            Label("Buy the movie", systemImage: "popcorn")
        }
    }
```

Put your function here



06

TabView

Its purely what’s on the right

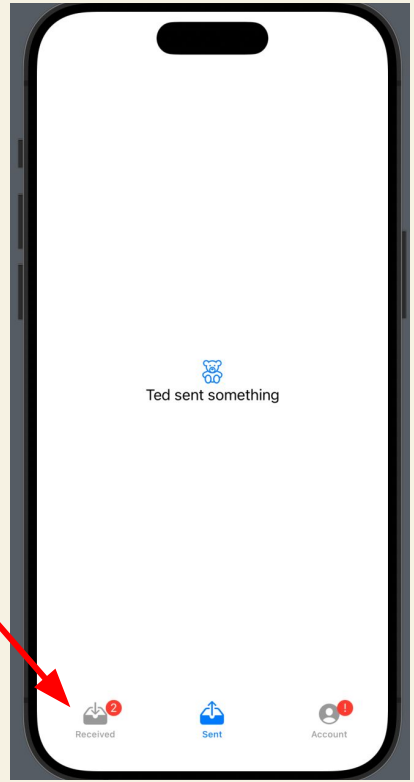
TabView - explaination

TabViews allow users to switch between different views more easily.

Applications:

- Settings page/view
- Account information

You can find TabViews almost on every iOS app!



TabView - code

First, you need to state NavigationStack, as the tab bar/TabView is part of navigation as it allows you to navigate to another view/page.

Secondly, declare TabView {...} to use tabs.

Lastly, declare the view that you want to navigate to (seen on the image in cyan. You need to state 'Tab' so it is a tab item, and give it a Text to show what view the user is going to.

Additionally, you can add a image using SF Symbols. You can see the list of the 30K symbols on the plus button in Xcode, and go to the page with a star embedded in a circle

```
import SwiftUI

struct TabberView: View {
    var body: some View {
        NavigationStack{
            TabView {
                Tab("Received", systemImage:
                    "tray.and.arrow.down.fill") {
                    ContentView()
                }

                Tab("Sent", systemImage:
                    "tray.and.arrow.up.fill") {
                    SomethingView()
                }
            }
        }
    }
}

#Preview {
    TabberView()
}
```

Mini Challenge

Make a few (<5 (less than 5)) tabs showing views of these themes below:

- **M**essaging app (WhatsApp, Telegram, etc)
- **A**nnouncement app
- **P**hotos App
- **S**ocial media

The views does not need to be complete or finished, but representing the theme of your app.

07

Homefun!

Finally it took me 4 hours to make this entire deck of slides like, what??? Why does it take so long. But it is for the sake of yall. :D

Home is fun

Today’s task, create a task manager that has these requirements:

- Has a list of tasks (e.g "do chinese hw")
- Have a toggle next to each task to mark it as "Done" or "Incomplete"
- Use a stepper to set the priority for each task (e.g from 1 to 3)
- Context menu that appears when you long-press on a task to delete the task and add a task (Use an array and add or subtract a task!)
- TabView with two tabs: one for "All Tasks" and one for "Completed Tasks."

Optional:
Use Spacer() to style your app!
Due next Monday/inc session!

No Blooket?

Its kahoot

Thanks!

Do you have any questions?

joshua_sheen@s2024.ssts.edu.sg
kesler_ang_kang_zhi@s2023.ssts.edu.sg

Feedback form

https://tinyurl.com/bdds8k2z

joshua_sheen@s2024.ssts.edu.sg
kesler_ang_kang_zhi@s2023.ssts.edu.sg